AFIT/GCS/ENG/99M-08

METHODOLOGY FOR APPLICATION DESIGN USING INFORMATION DISSEMINATION AND ACTIVE DATABASE TECHNOLOGIES

THESIS

Robert H. Hartz, Captain, USAF

AFIT/GCS/ENG/99M-08

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1999	Master's Thesis

4. TITLE AND SUBTITLE METHODOLOGY FOR APPLICATION DESIGN USING INFORMATION DISSEMINATION AND ACTIVE DATABASE TECHNOLOGIES	5. FUNDING NUMBERS
6. AUTHOR(S) Robert H. Hartz, Capt, USAF	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology 2950 P Street, Bldg 640 WPAFB OH 45433-7765	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GCS/ENG/99M-08
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Capt Terry A. Wilson/DDB Program Manager AFRL/SNAS 2010 5th Street WPAFB OH 45433-7001 COMM: (937)255-6329 x2634 DSN: 785-6329 x2634	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Dr Henry B. Potoczny
COMM: (937)255-3636 x4282 DSN: 785-3636 x4282
henry.potoczny@afit.af.mil

12a. DISTRIBUTION AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution unlimited.	

13. ABSTRACT *(Maximum 200 words)*

In dynamic data environments, the large volume of transactions requires flexible control structures to effectively balance the flow of information between producers and consumers. Information dissemination-based systems, using both data push and pull delivery mechanisms, provide a possible scalable solution for data-intensive applications. In this research, a methodology is proposed to capture information dissemination design features in the form of active database rules to effectively control dynamic data applications. As part of this design methodology, information distribution properties are analyzed, data dissemination mechanisms are transformed into an active rule framework, and the desired reactive behavior is achieved through rule customization. The methodology is applied to dynamic data test case scenarios to demonstrate the design of dissemination-based active rules. The results of applying the methodology to test case scenarios demonstrated that encapsulating information dissemination concepts into active rule structures could provide flexible database control strategies for dynamic data applications.

14. SUBJECT TERMS DATABASES, INFORMATION TRANSFER, INFORMATION SYSTEMS, DATA MANAGEMENT, METHODOLOGY, DESIGN, OODBMS, ECA	15. NUMBER OF PAGES 88
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

AFIT/GCS/ENG/99M-08

METHODOLOGY FOR APPLICATION DESIGN USING INFORMATION DISSEMINATION AND ACTIVE DATABASE TECHNOLOGIES

THESIS

Presented to the Faculty of the Graduate School of Engineering

Of the Air Force Institute of Technology

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Engineering

Robert H. Hartz, B. S. E.

Captain, USAF

March, 1999

AFIT/GCS/ENG/99M-08

METHODOLOGY FOR APPLICATION DESIGN USING INFORMATION DISSEMINATION AND ACTIVE DATABASE TECHNOLOGIES

Robert H. Hartz, B. S. E.
Captain, USAF

Approved:

Henry B. Potoczny

Henry B. Potoczny, Ph.D.
Thesis Advisor

March 5, 1999.
date

[illegible]

Michael L. Talbert, Maj, Ph.D., USAF
Thesis Committee Member

5 Mar 1999
date

[illegible]

Scott A. DeLoach, Maj, Ph.D., USAF
Thesis Committee Member

5 Mar 99
date

Acknowledgments

I would like to extend my sincere appreciation to the faculty members of my thesis committee. I wish to thank Maj Michael Talbert for his unwavering enthusiasm and guidance throughout this research effort, and I also want to acknowledge Maj Scott DeLoach for the insightful comments and support he provided to me. I would like to express a special thanks to my thesis advisor, Dr Henry Potoczny, whose patience and encouragement during my frequent research direction changes have enabled me to achieve a much deeper learning experience while at AFIT.

I would also like to acknowledge the contribution of the members of my sponsoring organization, the Air Force Research Laboratory's Sensor Directorate. I am sincerely thankful for the combined efforts of Capt Terry Wilson, Maj Tom Rathbun, and Capt Eric Baenen in providing me with useful insight and key documentation related to the Dynamic Database program.

Most importantly, I would like to express my grateful appreciation to my loving wife, Sandy, and my children, Bryan and Jennifer, for their continuous love, encouragement, and understanding throughout the past eighteen months. I offer my final thanks to the Lord, who is my true source of inspiration.

Table of Contents

List of Figures

AFIT/GCS/ENG/99M-08

Abstract

In dynamic data environments, the large volume of transactions requires flexible control structures to effectively balance the flow of information between producers and consumers. Information dissemination-based systems, using both data push and pull delivery mechanisms, provide a possible scalable solution for data-intensive applications. In this research, a methodology is proposed to capture information dissemination design features in the form of active database rules to effectively control dynamic data applications. As part of this design methodology, information distribution properties are analyzed, data dissemination mechanisms are transformed into an active rule framework, and the desired reactive behavior is achieved through rule customization. The methodology is applied to dynamic data test case scenarios to demonstrate the design of dissemination-based active rules. The results of applying the methodology to test case scenarios demonstrated that encapsulating information dissemination concepts into active rule structures could provide flexible database control strategies for dynamic data applications.

1 Introduction

Dynamic data environments are characterized by a collection of heterogeneous loosely-coupled data sources where a stream of information updates must be distributed to interested clients in a timely manner. The high velocity of changes to source data and derived data computations demand dynamic control structures that can efficiently tune system performance to the fluctuating environment. Satisfying information needs of numerous clients often requires finding a proper balance of data pull and push technologies to achieve the desired level of responsiveness.

The Dynamic Database (DDB) program [DDFA98] sponsored by the Defense Advanced Research Projects Agency (DARPA) is developing the capability to convert multi-sensor data into a responsive, comprehensive history of the sensed battlespace for warfighters. To provide ready access to sensor observations over time, the essential battlespace information must be efficiently stored and retrieved in a timely manner. A key challenge of this objective is the development of database technologies and distribution services for efficiently managing, querying, updating, reporting, and processing the large volume of sensor data that is transformed and aligned on a global common schema.

1.1 Problem Statement

In data-intensive applications, the amount of data processing activity and resource consumption associated with application tasks can cause significant performance degradation. Data dissemination-based system concepts can provide a mechanism to achieve scalable hierarchies of information flows among data producers and consumers. In addition, efficient dynamic structures are necessary to effectively control the large volume of data transactions in a consistent manner. Encapsulating information dissemination features within an active rule framework can provide embedded application control strategies as well as performance improvements for a dynamic data environment.

1.2 Research Goals

The goal of this research is to provide a methodology to assist developers in designing data dissemination features for an application using automated database controls. Through this methodology, a process for analyzing and classifying the information distribution properties of an application is introduced. In addition, the transformation of data dissemination features into an active database rule framework is developed as a research sub-goal. A process is also developed to analyze and customize the remaining active rule dimensions to achieve the desired level of reactive behavior. Finally, this research will demonstrate, through a case study, that the methodology can be used to design dissemination-based active rules for a dynamic data application.

1.3 Assumptions

The analysis techniques used in the active dissemination-based design methodology do not presume any particular database or application design approach is in use. In addition, general-purpose active database rule syntax and generic dissemination communication services are used throughout this research to express the methodology concepts and case study test cases. The examples shown in this research are based on an object oriented data model, but most of design principles of the methodology would equally apply to either a relational or object-relational data model.

Since the objective of this research is primarily focused on the dissemination-based active rule design, no implementation-specific issues are addressed, nor have any performance studies been performed on the resulting rule structures. Finally, this active rule design methodology does not conduct any static rule analysis for termination or confluence properties at this time.

1.4 Document Organization

Chapter 2 introduces the event-condition-action (ECA) rule definition model, rule execution semantics, and implementation challenges of active databases. Also, the features of dissemination-based information systems and associated design tradeoffs are considered in the background chapter. Chapter 3 details the steps used in the active dissemination-based design methodology. The first step examines the data dissemination characteristics to be captured in a dynamic design model. The actual conversion of data distribution concepts into an active rule framework is presented next in this methodology. In the last methodology step, the active rule components are tuned to produce the proper behavioral semantics. In Chapter 4, an evaluation of the methodology is performed using components of a dynamic data application as a test case. Finally, chapter 5 summarizes the findings, recommendations, and future directions for this research.

2 Background

In this chapter, some background material is provided on active database and information dissemination technologies. As part of an active database orientation, rule definition features, rule execution semantics, and implementation issues are covered. Also, the essential features and design considerations associated with information dissemination systems are presented.

2.1 Active Database Technology

Active database systems combine the data storage capabilities of passive databases with the reasoning of artificial intelligence rule-based systems to dynamically perform actions in response to selected data manipulation or user-defined events. By enhancing passive database management systems (DBMS) with rules, active databases can efficiently perform functions previously encoded in application code, accomplish tasks allocated to internal database sub-systems, and facilitate new applications beyond the scope of passive databases [WC96].

It is difficult to categorize specific characteristics required of all active database systems without considering the intended application domain. Although many active database applications are still primarily research prototypes or single domain applications, there are some general features common to most active DBMS implementations [PDW+93][FT95][DGG96]. The key characteristics of active databases examined in this section are rule definition, rule execution, and implementation issues.

2.1.1 Rule Definition

By embedding active rules into databases, many of the behavioral semantics normally present in an application program can be formally expressed with data rule definitions. Since there are two types of rules associated with databases, the similarities and differences of active and deductive rules will initially be examined. Then, the syntactical elements of the most common form of active rules, event-condition-action rules, will be described.

2.1.1.1 Active and Deductive Rules in Databases

Database rules are distinguished as two different types: *deductive rules,* which are used to express knowledge about application domains in a purely declarative way; and *active rules,* which are used to express actions to be performed in response to events that may be internal or external to the system [FWP97].

Although both active and deductive databases achieve a level of knowledge independence from applications, the semantics of their rules are quite different [CR96]. Deductive rules express declarative knowledge. Influenced by artificial intelligence work in logic programming, deductive rules represent queries in a style that describes the meaning of the query (the *what*) and does not depend on the query evaluation strategy (the *how*). In contrast, active rules, with roots in rule-based expert systems, express knowledge in a procedural form through event-condition-action computations. The behavioral semantics of active rules are based on reactive computations that occur automatically in response to data manipulation or other registered events. In one study of relational database rule language differences, highly abstract deductive rule languages lie on one end of a spectrum, while active rule languages possessing more powerful constructs are on the other extreme, with rule languages of varying degrees of expressiveness in between as shown in Figure 1 [Wid93].

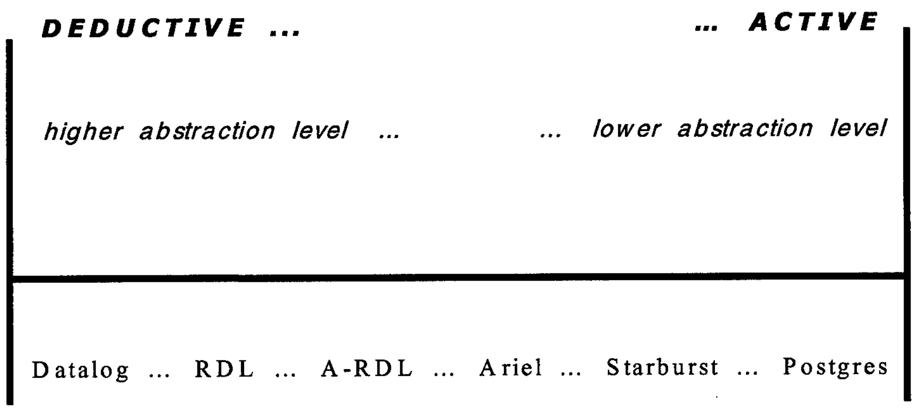
DEDUCTIVE ...
... ACTIVE
higher abstraction level ...
... lower abstraction level
Datalog ... RDL ... A-RDL ... Ariel ... Starburst ... Postgres

Figure 1: Spectrum of Database Rule Languages

Although the syntax and semantics of deductive databases are fairly standard, active database systems have no widely accepted formal characterization. There is broad agreement on syntax and semantics of active rules; however, recent research work has focused on building formal foundations for active rules [FWP97]. Other significant active DBMS research has proposed using static analysis algorithms to ensure active rule processing guarantees termination, confluence, and observably deterministic behavioral properties [AHW95][KC95][BCP96].

2.1.1.2 Event-Condition-Action (ECA) Rule Model

The most common type of active rules is *Event-Condition-Action (ECA)* rules [DBB+88], meaning "WHEN a current event occurs, and IF a given condition holds, THEN a certain action is executed." ECA-rules are also known synonymously as *triggers* [HAC+97], *alerters* [GFV96], *production rules* [Pat95], and *situation-action rules* [DBB+88]. Active database rule definitions specify the events, conditions, and actions used by the ECA-rule model, as well as any rule groupings and priority relationships among rules [WC96]. Since the operational semantics of ECA-rules is independent of the underlying data model, active databases with similar functionality have been built over both relational and object-oriented databases [FWP97]. The detailed features of an ECA-rule may best be illustrated by looking at an example of the Starburst active database language in Figure 2 [Wid96]:

create rule *name* **on** *table*
when *triggering-operations*
[**if** *condition*]
then *action-list*
[**precedes** *rule-list*]
[**follows** *rule-list*]

Figure 2: Create Rule Command in Starburst

The *name* in Figure 2 identifies the rule, and *table* is the name of the relational database table over which the rule is defined. With an active object-oriented data model, ECA-rules are declared as first-class objects with attributes and methods, and inheritance and aggregation properties can be used to build rule hierarchies [DBB+88].

The **when** clause of Figure 2 identifies the event that causes the rule to be triggered. An ECA-rule event type can be *primitive,* defined as an elementary occurrence of interest, or *composite,* a combination of primitive or composite events separated by logical constructors such as sequence, conjunction, or disjunction [DGG96]. Primitive database modification operations such as insert, delete, and update are internal events that can cause rules to be triggered, but other events such as time events, transaction events, method-invocation events, or events originating from a source external to the database may also trigger a rule [PDW+93][BZBW95]. Some rule events may contain parameter data that can be used for the rule's condition evaluation and action execution, if necessary [GD93].

In Figure 2, the **if** clause specifies a condition to be evaluated once the rule is triggered. In this example, the optional condition clause specifies some state of the database to be evaluated, and if omitted, a variant of ECA-rules called event-action rules is produced with an assumed true condition [Wid96]. A rule condition is satisfied if a predicate on the database state evaluates to true or the result of a specified query is non-empty [DHL90].

The **then** clause of Figure 2 defines the list of actions to be executed once the rule is triggered, and the specified condition evaluates to true. At a minimum, the action list may contain data modification or retrieval operations, but database transaction operations or application subprograms and methods may also be part of the action list [DGG96]. Unless scheduled concurrently, the actions specified are executed sequentially in the order defined in the rule [WC96].

Finally, the **precedes** and **follows** clauses shown in Figure 2 are optional and used to specify priority ordering between rules. The defined rule will execute before any rule in the **precedes** list and after the rules in the **follows** list, if triggered at the same time; however, cycles in the rule priority ordering are not permitted [Wid96]. Assigning similar priorities to a group of related rules or using unique numeric priority values for each rule are alternate ways a rule designer can specify the firing order of concurrent rules to the execution model [FT95][WC96].

Rules may be classified as either instance-oriented or set-oriented depending on whether rules react separately for each distinct updated item or are triggered by a collective modification and activated only once [FT95]. For collective modifications, rule conditions and actions may act on special logical entities that contain all the inserted, deleted, or updated database tuples resulting from the rule's triggering event [Wid96]. Finally, ECA-rule definitions may include rule-structuring features for organizing individual rules into designated collections or sets [WC96].

2.1.2 Rule Execution

Besides rule syntax, the semantics of rule execution must also be consistent among active database implementations. An execution model is the primary feature necessary for active database rule processing. Detecting event occurrences, evaluating conditions, and coordinating action execution with database transactions are all responsibilities of the execution model [DGG96].

2.1.2.1 Event Handling

Active databases have mechanisms to detect all structured query language (SQL) data modification events automatically, but all active DBMSs may not detect primitive events such as temporal events, method events, and external application events [DGG96]. For temporal events, an absolute time event may be signaled after a system clock interrupt, and relative time events require the system to monitor a specified time interval for event detection [GD93]. For signaling method events, *method wrapping* brackets a method with both a begin-method and end-method signal [Buc98], and active DBMS implementations include wrapping all methods automatically [BZBW95], only wrapping methods of special object classes [AC95], or manually adding wrappers as needed [GD93]. To signal events external to the active DBMS, users or applications must notify the system by using an explicit operator to raise the event [GD93]. Composite events for active rules are composed of primitive events and algebraic operator combinations [DHL90]. Specialized data structures, such as colored Petri nets [GD93], are sometimes used to detect complex composite event patterns, and active DBMS implementations must support a persistent event history for composite events that span several database transactions [DGG96].

During rule processing, the rule execution model must also control the treatment of triggering events, known as consumption. Event consumption issues include whether processed events retain their capability of triggering more rules (scope of event consumption), and when does actual consumption take place for multiple occurrences of the same event type (time of event consumption) [FT95]. There are three ways to deal with the scope of event consumption: no consumption, local consumption, and global consumption. If no consumption is used, triggering events retain their capability to trigger rules, similar to production rules, until the condition becomes false. Using local consumption, the triggering event may activate other rules, but not the processed rule, and finally, global consumption restricts any additional rules from being triggered. With event consumption timing, most active rule implementations support recent mode, where the latest event is consumed, and chronicle mode, where events are consumed in chronological order, although other event consumption policies are possible [BZBW95].

2.1.2.2 Condition Evaluation

Condition evaluation can be a large performance bottleneck for active database systems with a large rulebase. Three techniques for efficiently evaluating rule condition are identification of common sub-conditions, materialization of intermediate results, and incremental condition evaluation [DBB+88]. Artificial intelligence discrimination networks, such as Rete and TREAT, are sometimes used in active databases to reduce the condition evaluation performance overhead [Pat95][HBC+97]. In addition, another body of active DBMS research [BCL98] introduces the concept of logical events to limit the number of unnecessary ECA-rules triggered, subsequently reducing the amount of extraneous condition evaluations to be performed.

The condition evaluation of the execution model must also possess the capability to pass parameter information from events to conditions, and the condition must also be able to refer to data items bound after event detection [DGG96]. Likewise, action execution must have access to relevant information from events and conditions.

2.1.2.3 Coupling Modes and Transactions

As part of rule execution, active databases must offer different synchronization strategies between event detection, condition evaluation, and action execution called coupling modes [DHL90]. With both immediate and deferred coupling, *triggered* transactions are treated like sub-transactions of the *triggering* transaction [DGG96]. Immediate coupling occurs when a triggered transaction is executed directly after the triggering transaction is detected, and deferred coupling happens when a triggered transaction is executed at the end of the triggering transaction, but before it commits. In the detached mode, the final coupling approach, the triggered transaction is started as a separate transaction and is independent from the triggering transaction. An example of the three modes used for event-condition coupling is illustrated in Figure 3 [SUC98].

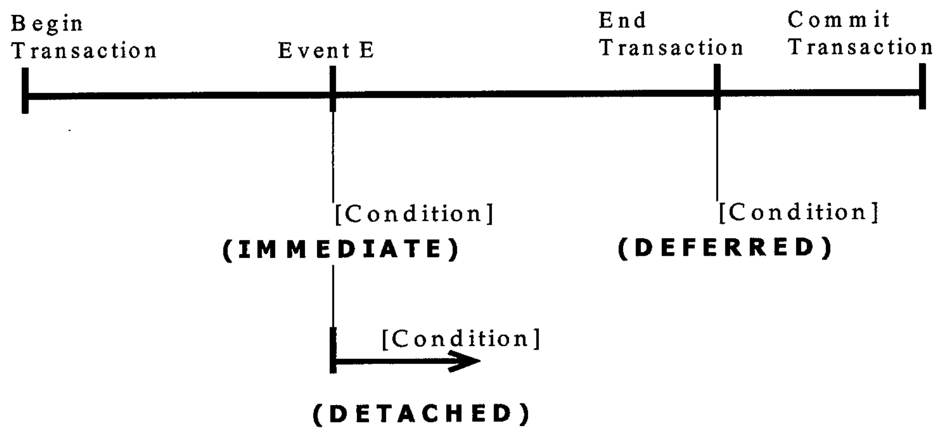

Figure 3: Basic Coupling Modes between the Event and Condition

Since there are many combinations of event-condition and condition-action coupling modes, several performance studies have been conducted with active databases using different transaction models. For lightly loaded systems, the type of coupling mode used had little impact on response time [CJL91]. Active DBMS coupling performance experiments have focused on different active workloads [CJL91][PDW+93][AKGM96], including real-time systems

[PSS93][Ulu98][SST96][SUC98], and various transaction models [DHL90][AC95][CA95][MT95][Ulu98].

2.1.2.4 Miscellaneous Rule Execution Features

The semantics of active DBMS rule execution should also prescribe the behavior of rule conflict resolution strategies and data binding modes. The execution model's conflict resolution strategy for multiple rules may include serial or parallel execution [FT95]. With a serial approach, the rule designer may assign priorities to help resolve conflicts (e.g., Starburst's **precedes** and **follows** clause [Wid96]). For concurrent execution, all triggered rules in the conflict set are executed in parallel by scheduling separate condition evaluation/action execution transactions for each rule.

Execution models also support *instance-oriented* or *set-oriented* binding modes for determining the granularity of the event as well as the data items that the condition and actions, associated with the event, can access [PDW+93]. When the binding mode is *prior,* conditions and actions can also refer to the value of a data item just before the event was detected.

2.1.3 Active Database Implementation Issues

Having outlined the rule syntax and execution semantics of active databases, there are still some issues that designers should consider when implementing applications with active functionality. This section will discuss the tradeoffs of different architectural approaches, present categories of active DBMS applications, and overview some future areas of active database research.

2.1.3.1 Architectures

The primary distinction between active database architectures is the level of integration between the passive database and the active components. The three architectural approaches examined in this section are integrated, layered, and unbundled architectures.

Integrated architectures can either be achieved by adding active features to an existing passive DBMS or by building an active database system from scratch [Cha92][WC96]. The advantages of tightly coupling active components with the underlying database are a wider range of functionality and more efficient performance, but the main drawback is the substantial development effort in either creating or modifying complex DBMS code to accommodate active features [Buc98]. The REACH active DBMS uses an integrated architecture with Texas Instrument's Open OODB system [BZBW95].

A layered architectural approach of implementation builds the active functionality on top of an existing passive DBMS, requiring a communication layer between the DBMS and the active components [WC96]. Although performance can suffer by not being able to interact directly with internal DBMS subsystems, layered architecture benefits include lower implementation costs and possible reuse of the same interface for multiple passive databases [Buc98]. The SAMOS active DBMS prototype is implemented using a layered approach on top of the ObjectStore commercial DBMS as shown in Figure 4 [GGD+95].

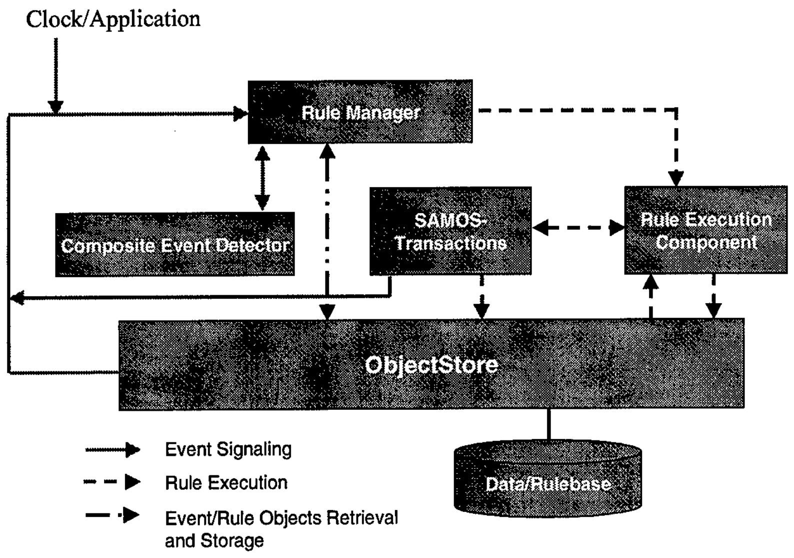

Figure 4: Architecture of SAMOS Kernel on top of ObjectStore

Although most active functionality is bundled together with passive database systems, some researchers feel the active capabilities should be uncoupled from the continually increasing functionality included in most passive DBMSs [GKVB+98]. Some advantages of stand-alone active functionality are applications not requiring databases may also use active rules, heterogeneous information sources are easier to include into active applications, and unbundled active mechanisms may more easily port to other DBMSs. Some remaining challenges with having active functionality separated from the database are whether the full active behavioral semantics can exist outside the DBMS and the lack of mature active rule services in open architecture environments.

2.1.3.2 Application Classifications

The power and versatility of active rules make active databases well suited for a variety of applications. Two factors that can be used to classify active DBMS applications are the role the active functionality plays within the information system (monitoring or controlling) and the degree of information integration (homogeneous or heterogeneous) [DGG96]. Using combinations of the information system role and integration level, active DBMS features can be grouped into three meaningful application classes: monitoring in a homogeneous system, controlling in a homogeneous system, and controlling in a heterogeneous system. In addition, a study has been performed to determine which common active database features are ideally suited for application domains such as integrity constraint checking and derived data updates

[PDW+93].

Active databases have been used to automatically enforce data integrity constraint errors caused by data manipulation events, maintain consistency between source data and summarized views of data warehouses, and support business policies in applications by means of 'business rules' [Wid96]. Some other prototype active database applications include workflow management systems [DHL90], navigation systems [PSS93], manufacturing control [LRST93], battlefield awareness [DSLL97], network services [PSS93], and data mining [HNK94].

2.1.3.3 Future Directions

Active databases are powerful mechanisms for creating 'knowledge independence' from applications by expressing the appropriate domain semantics in the form of rules, but there are still active database issues needing further research. Future work needs to continue to make active database rule syntax and semantics more standard and interoperable with more formal representations [FT95][FWP97]. In addition, further research should be conducted to examine better ways to uncouple active functionality from monolithic databases, yet maintain the same desired level of reactive behavior through the use of distributed active services [GKVB+98]. Finally, active database research should address more issues related to real-time and temporal databases, like data deadlines [RSS+96] and timeliness requirements [SST96][XSR+96].

If active databases are to migrate from research prototypes to general-purpose applications, a usable suite of development tools should be available for domain users and developers. To achieve maximum benefit from active DBMS implementations, programming environments should contain the following functionality as separate tools or extensions of existing tools: rule browser, rule designer, rulebase analyzer, a debugger, a maintenance tool, and a performance tuning tool [DGG96].

2.2 Data Dissemination

With the continued advancement of communications technology and proliferation of information available on the Internet, the demand for dissemination-based applications that can harness information flows within data-intensive environments is growing. These data dissemination applications use controlled delivery mechanisms to move data from a set of producers to another, typically larger, set of consumers [FZ96]. Properly configured data dissemination systems prevent information overload by balancing data push and pull requirements without enduring a large performance penalty. As an overview of dissemination-based information systems, the essential architectural components, design considerations, and data delivery mechanisms are presented in the following sections.

2.2.1 *Primary Architecture Components*

Although the implementation details of applications may vary, dissemination-based systems are largely designed around three essential architectural elements: data sources or *producers*, clients or *consumers*, and information *brokers* [FZ97]. These components may be hierarchically designed in an information processing chain for some data-intensive domains. For example, a particular system node might simultaneously be considered a consumer of upstream data, a broker that transforms the retrieved data, and a producer for any downstream activities.

2.2.1.1 Data Sources (Servers)

Data sources, also known as servers in dissemination-based systems, are the origins of raw information that is to be disseminated. The underlying content of these heterogeneous information sources may be in many different formats, to include: unstructured text, semi-structured Web documents, images, stream-based multimedia information, and structured data from database management systems [LS97]. Data servers may passively retrieve information in response to a user request, or a server may actively transmit data to clients based on predefined user interests or upon source data modification. To help clients retrieve relevant data, research efforts are focusing on using metadata access and domain ontologies to better describe source data contents [YA95][LS97][RS98].

2.2.1.2 Clients

Dissemination-based information systems usually have a large population of clients relative to the number of data sources [FZ96]. In a case study of Web-based dissemination applications [FZ97], client data requests were characterized as fairly small, focused primarily on new or recent changes to data, and contained a great degree of overlap among user interests. To reduce overall data latency, information consumers are relying on push-based user profiles [YGM95] and caching strategies [YA95] in dissemination applications.

2.2.1.3 Information Brokers

As central elements of a data dissemination application, the information brokers are responsible for collecting producer data, making any enhancements to the data, and distributing the information to consumers [FZ97]. Unlike the common features among producers and consumers, brokers encapsulate a variety of different functionality within dissemination-based systems, and depending on the implementation, these intermediary elements may be known as information brokers [FZ97], mediators [LS97], agents [CB97], gestalts [RS98], or filtering engines [YGM95]. Filtering retrieved information, locating user data requirements, semantically structuring or organizing data, and notifying users of significant events are some of the important tasks performed by brokers [CB97]. The task breakdown can be distributed to a hierarchy of intercommunicating brokers depending on the size and functionality of the application [FZ97].

2.2.2 *Design Issues*

Several key design issues of the information environment must be considered to achieve a scalable, customized dissemination-based solution. The three main factors of dissemination systems that must be analyzed are the primary direction of data flows, the timing of data delivery, and the type of communication protocol used [FZ96]. In addition, selection of intelligent client profile management schemes and effective data caching strategies can also yield system performance improvements.

2.2.2.1 Server Push vs. Client Pull

Passive data servers have traditionally been 'pull-based', where information is transferred to a client after a request has been initiated. On the other hand, 'push-based' data delivery sends information in advance of any specific client requests [FZ97]. Some drawbacks of a data pull approach include:

server contention, a priori knowledge of data requirements, and limited flexibility for scheduling data delivery. However, push-based approaches can result in network bandwidth congestion, fail to accurately predict client data requirements, and are targeted primarily for new or recently updated data. Therefore, the cost of initiating a data transfer and the precision of client data requirements are important criteria for selecting the types of data delivery mechanisms [FZ96].

2.2.2.2 Periodic vs. Aperiodic Processing

Data push or pull can be performed in either a synchronous or asynchronous manner. Periodic delivery is conducted according to some repeating schedule [FZ97], and it is best suited for situations where clients may be unavailable (e.g., mobile users) [DMS97] or must meet real-time timing constraints [BB97]. In the design of a periodic system, polling too frequently increases performance overhead, while infrequent polling can lead to data staleness [FZ98]. In contrast, aperiodic delivery is triggered by an event such as a client action (information pull) or a data update (information push) [FZ97]. Clients that consistently monitor data communication for updates [DSLL97] or can tolerate missing information benefit the most from event-driven information dissemination [FZ96].

2.2.2.3 Unicast vs. 1-to-N Communication

The third major design consideration is whether data delivery mechanisms employ unicast or 1-to-N communication. With unicast communication, information is sent using a point-to-point connection between a data source and one other machine [FZ96]. Data dissemination systems use 1-to-N communication in two different ways: multicast and broadcast. With a multicast protocol [Gla96], data is sent to a selected group of clients, who have previously declared interest in the information, while data transmitted in a broadcast mode can be received by an unknown and unbounded set of clients [FZ97]. Since developed network protocols can guarantee the eventual delivery of data to an authorized client, unicast and multicast approaches are considered reliable forms of communication [FZ98]. Scalability of multicast or broadcast communication can be achieved by using local server nodes to handle all dissemination traffic for an organization with a commonality of interest [YGM94] [DMS97].

2.2.2.4 Profile Management

For data push applications, an accurate representation of a user's information interests, known as a profile, allow data sources to better anticipate the data requirements of a client. Good profiles should minimize the amount of relevant information that is missed, and reduce the number of irrelevant data items retrieved [YGM95][CB97].

Profiles have been implemented as continuously executing queries [TGNO92], collections of tables [RS98], statistically weighted vectors [YGM95], and boolean predicates [YGM95][LEF98]. Some research initiatives for improving profile-matching performance include: using AND-OR graphs of predicates for efficient profile evaluation [LEF98], extracting profile data from overlapping queries [DFJ+96][CBGM98], and saving bandwidth by grouping common profile interests [YGM95][SDSV97].

2.2.2.5 Caching

Caching strategies in dissemination-based systems should be closely integrated with the design decisions for data flow direction and timing to achieve peak performance. Although caching is similar in most ways to other applications, dissemination systems offer a few design challenges. Implementation policies must decide whether cached copies of recently modified data will be propagated or invalidated among a multitude of potential clients [FZ97]. Research on cache replacement policies for data dissemination systems include semantic locality replacement based on query access patterns [DFJ+96] and cost-based replacement for broadcast disks [AFZ96]. Imagery and multimedia objects, also known as heavyweight objects, present a unique challenge for bandwidth allocation in dissemination-based systems. One application transmits metadata for the heavyweight object to the client, allowing the user to assess the relevance of the heavyweight object prior to disseminating it [SDSV97].

2.2.3 *Delivery Mechanisms*

The different types of data delivery mechanisms implemented in dissemination-based systems are shown in Figure 5 [FZ98]. In the following sections, four classifications of mechanisms are described: aperiodic pull, periodic pull, aperiodic push, and periodic push.

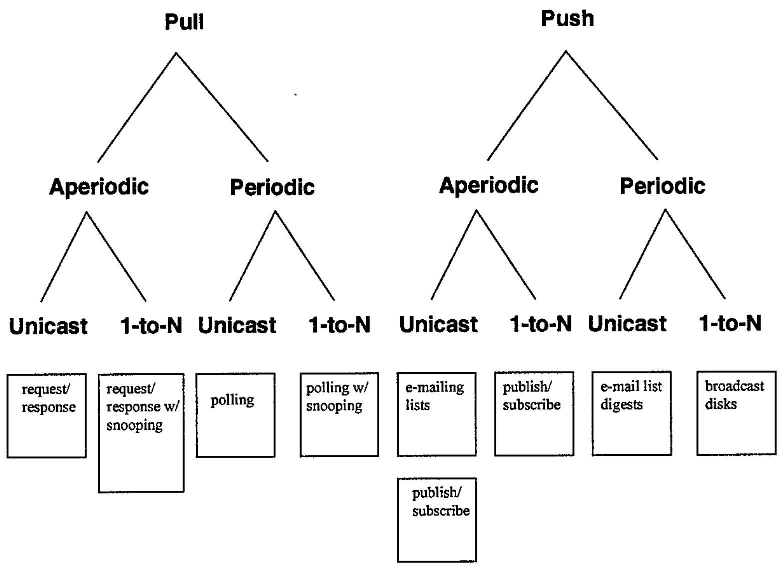

Figure 5: Data Delivery Options

2.2.3.1 Aperiodic Pull

With a unicast connection, this data delivery option is the traditional client request/server response mechanism. When aperiodic pull is used with 1-to-N communication, the mechanism is characterized as 'request/response with snooping' since some clients may obtain data they did not explicitly request. These mechanisms can exhibit scaling problems since the rate a server can handle pull requests is fixed and as the number of requests grows, data latency also increases [AFZ97].

2.2.3.2 Periodic Pull

Periodic pull mechanisms are used in applications to obtain information or status from remote data sources on a regular cycle. Both unicast and 1-to-N communication are considered polling data delivery options, but polling with the 1-to-N connection can also snoop to retrieve data not requested [FZ98].

2.2.3.3 Aperiodic Push

As an increasingly popular way to disseminate data, aperiodic push delivery alternatives are also known as publish/subscribe protocols [YGM95][Gla96]. Most push-based publish/subscribe mechanisms communicate to multiple clients, but some e-mail list mechanisms or database triggers use a unicast connection for implementation. These protocols are ideally suited for dynamic source data that can be pushed to clients based on their respective user profiles [RS98].

2.2.3.4 Periodic Push

An example application of periodic push data delivery using a unicast connection is an email list that collects and forwards digest updates on a regularly scheduled cycle, so a user is not continually interrupted with messages. A more common periodic push delivery option is called *broadcast disks* [AFZ95] that use 1-to-N communication links.

A broadcast disk implementation continuously and repeatedly sends data on a broadcast channel that a client can access. By broadcasting data at different frequencies based on the interests of clients, the broadcast channel emulates multiple disks of different sizes and speeds from a user's perspective [AFZ95]. Figure 6 [AFZ97] shows an example of a broadcast program emulating three disks with relative spinning speeds of 4:2:1, and data unit **A** is on the fastest disk, units **B** and **C** are on the medium disk, and units **D**, **E**, **F**, and **G** are on the slowest disk. Broadcast disk programs have been an active research topic in dissemination-based information systems, and recent work has examined broadcast disk scheduling [AF98], caching [AFZ96][AFZ97], push/pull bandwidth allocation [AFZ97], and real-time constraints [BB97].

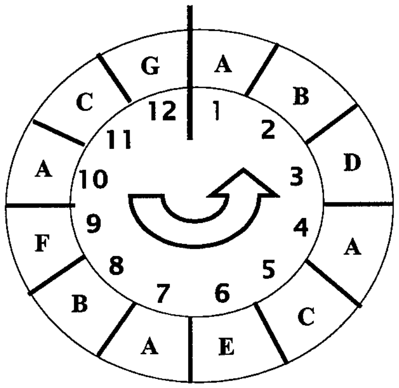

Figure 6: Example of a 7-Unit, 3-Disk Broadcast Program

2.3 Summary

This chapter provided an overview of the syntax and semantics of active database rules, and the design components and classification of information dissemination mechanisms were also covered. In Chapter 3, a methodology will be introduced to convert the inherent design characteristics of information dissemination applications into the essential active rule features.

3 Methodology

Designing applications with active database mechanisms is not a widely practiced endeavor, and therefore, well-established design methodologies for active rule implementation are not very prevalent. Since active functionality is typically layered between the database and the application, active rule design approaches have ranged from extending the passive database schema design [NTM+95], to extracting behavioral semantics of an application into a modular set of partitioned rules called stratification [BCP96][CF97]. While this research complements other database schema and application design approaches, the primary focus of this proposed methodology is to build dynamic structures to intelligently control the flow of data between producers and consumers.

This chapter presents a methodology for designing data dissemination-based mechanisms into a dynamic data application through the use of active database rules. The objective is to help application designers capture the semantics of information distribution policies for data-intensive environments in consistent and maintainable structures residing closely with the data.

The design process of the methodology is structured in three phases as shown in Figure 7: application analysis, rule transformation, and rule customization. Information distribution properties are mapped into dissemination classes in the application analysis step. Using the dissemination classes as input, the rule transformation phase specifies the events, conditions, and actions in an active rule framework. Miscellaneous rule features are added to the rule template to achieve the desired level of reactive behavior as part of the rule customization step. The resulting active rules generated by this methodology are general-purpose abstractions that can provide consistent behavior for a variety of dynamic data streams.

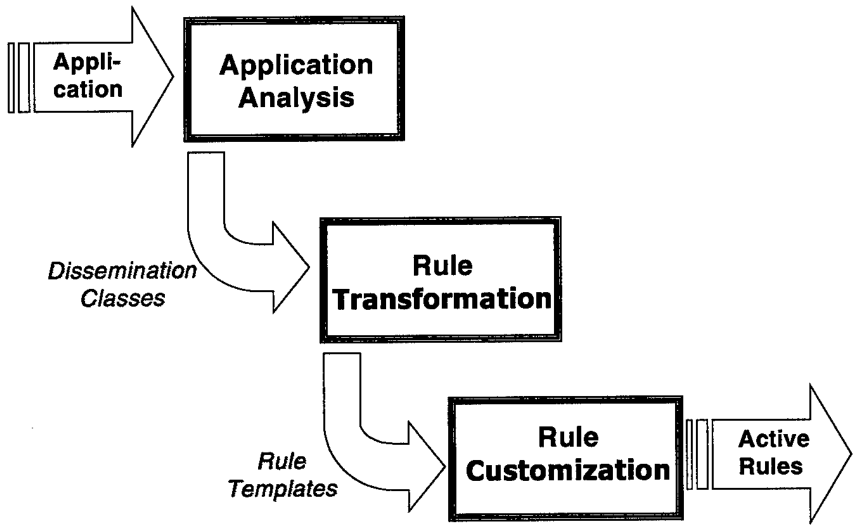

Figure 7: Design Methodology Steps

3.1 Data Dissemination Application Analysis

The first step of the methodology is to assess certain features that impact data distribution strategies for the primary data flows associated with application processes or threads. In this analysis, each relevant data processing scenario is evaluated in terms of activation mechanisms, data precision, data size, and client communication protocols. In the following sections, each of these important characteristics is examined in the context of high-volume data distribution.

3.1.1 Activation Mechanisms

When analyzing dynamic data processing scenarios, it is important to determine what type of event initiates the data processing activity. The activation mechanism used for a data flow process can impact the scheduling of transactions as well as the timeliness of the data. Active mechanisms for dissemination-based processing are classified as either *data-driven* or *time-driven* events [PDW+93].

Data-driven events are triggered by modifications to the affected data object. The data change can be initiated by data manipulation operations (e.g., insert, delete, update) or object methods that modify data attributes. Data retrieval operations such as query functions or methods accessing data attributes are also considered data-driven activation mechanisms. Data-driven triggers are useful when 'on-demand' processing is appropriate.

On the other hand, time-driven activation mechanisms are based on the temporal events associated with a system clock or calendar. Temporal triggers can be explicitly defined as absolute time events with optional repeating cycles, or time events can be specified implicitly to occur at some time interval relative to a specific event occurrence [DG93]. Time-driven activation mechanisms are often used to synchronize processing with periodically updated data sources or for selectively scheduling transactions to improve system performance.

3.1.2 Data Precision

Data precision, in this dissemination-based design methodology, refers to the degree of data correctness that must be obtained when processing transactions. For some data processing tasks, *exact* results must be retrieved at all times, yet semantically close matches or *inexact* objects may be sufficient for transactions involving uncertain data.

To achieve exact responses from complex data requests, query developers must have detailed knowledge of the underlying database schemas in advance. Exact results are also desired when performing aggregate computations over all instances of data objects. If exact results are required for temporal data items, transactions must ensure data elements remain valid or are refreshed before the transaction commits.

Inexact data results are preferred when clients are not entirely certain of their data requirements. User profiles are commonly use to describe client data interests, and data sources attempt to disseminate information that matches or nearly matches the profile. To effectively use inexact data distribution, data servers should monitor client statistical feedback of previous data submittals, and clients must adapt their profiles to meet changing data interests.

3.1.3 Data Size

Based on the availability of system resources, or by desire of the clients, the granularity of a server response to data transactions may include *lightweight* objects, such as notifications or data references, or the complete data components known as *heavyweight* objects [SDSV97].

Disseminating lightweight notifications or alerts usually occurs in response to some existential query condition or to signify some type of processing exception. As another lightweight object example, metadata transmissions typically are used when communications bandwidth is limited and large data objects such as images or multimedia files are involved. When only data descriptions or references to object identifiers are disseminated, clients can reduce some of the uncertainty associated with their requirements before formally requesting the actual heavyweight objects in question.

When heavyweight data objects must be distributed as part of a transaction, client cache strategies can be used to enhance system performance. In addition, large image or multimedia objects may require compression to minimize transmission latencies in constrained communications bandwidth environments.

3.1.4 Client Communication

The final data design consideration for this dissemination-based methodology is whether the data is distributed to individual or multiple clients. The client communication requirements can be categorized as either *unicast* for a single transmission or *multicast* for data delivery to a group of clients [FZ96].

Unicast data responses earmarked for an individual client typically are very specific in nature with constrained domain conditions. Unicast data communication is also appropriate when data is sensitive, and the authorized receipt of the information must be verified. However, frequent use of unicast data traffic can saturate communications networks and degrade overall system performance.

In high velocity data systems with numerous clients, there are frequently overlapping data requirements that can be multicast to different groups of clients to conserve communications bandwidth. If confirmation of data delivery is not required, broadcasting information instead of using multicast protocols can be an efficient dissemination approach, especially for mobile clients.

3.1.5 Feature Classifications

The mapping between the design methodology's application features and the recommended data dissemination delivery approach is shown in Figure 8. The two primary features for determining the general data delivery classification are the type of active mechanism used and the level of data precision required. However, data size requirements and client communication issues are important distinguishing characteristics for the detailed design of the active dissemination-based structures.

Activation Mechanism	Data Precision	Data Size	Client Communication	Recommended Data Dissemination Approach
Data-Driven	Exact Data	Heavyweight Objects	Unicast	Aperiodic Pull
			Multicast	
		Lightweight Objects	Unicast	
			Multicast	
	Inexact Data	Heavyweight Objects	Unicast	Aperiodic Push
			Multicast	
		Lightweight Objects	Unicast	
			Multicast	
Time-Driven	Exact Data	Heavyweight Objects	Unicast	Periodic Pull
			Multicast	
		Lightweight Objects	Unicast	
			Multicast	
	Inexact Data	Heavyweight Objects	Unicast	Periodic Push
			Multicast	
		Lightweight Objects	Unicast	
			Multicast	

Figure 8: Application Features Mapped to a Dissemination Approach

3.2 Transforming Dissemination Features to Active Rules

Once the data dissemination application features have been assessed, the next step in the methodology is to transform those features into the primary active E-C-A rule components: events, conditions, and actions. For each rule component, representative elements that could be specified in rule definitions are identified, and no specific active rule syntax is assumed. The process for transforming data features into an active rule framework will be described for the four dissemination-based classifications: aperiodic pull, aperiodic push, periodic pull, and periodic push.

3.2.1 *Aperiodic Pull*

Asynchronous data retrieval processes are typically single-client request-response transactions as indicated in the background chapter. Either heavyweight data objects or just the data references are retrieved, depending on the client caching capabilities or available network bandwidth. Multiple clients may also gain access to data by snooping through results that were disseminated to a common client repository in response to an individual request. The essential rule components for aperiodic pull data components are discussed in the following sections.

3.2.1.1 Events

The primary trigger event for aperiodic pull scenarios would be a data retrieval operation, such as *query()*. Depending on the application, a data request could also be initiated by a user method invocation. Finally, a data manipulation event on a data source may also trigger an aperiodic data request for downstream nodes before performing derived data computations.

3.2.1.2 Conditions

In some aperiodic pull scenarios, the condition predicate would be evaluated to true implicitly for the query processing. The query constraints for the data retrieval can be evaluated as the rule condition. Some safeguard conditions may also be implemented in rules to improve performance. For temporal data objects, the condition could block queries until data timestamps have expired to prevent redundant refresh transactions.

3.2.1.3 Actions

For data requests invoked by a user or application event, saving the query results in a transient collection for dissemination is an appropriate action. When hierarchical information nodes are involved, high-level queries in rules may be re-written with the necessary sub-queries included in the action list. Actions might also include appropriate calls to communications services to cache the retrieved objects or distribute the associated metadata to clients.

3.2.2 *Aperiodic Push*

Data dissemination techniques using aperiodic push are known as publish/subscribe protocols. Publish/subscribe delivery mechanisms are ideal for dynamic data environments or temporal data items since new data is constantly being propagated to match client profiles. Profiles can be developed so lightweight metadata or messages can serve as indicators to inform clients of any unusual data activity. Although publish/subscribe techniques can keep heavyweight objects continuously up-to-date, pushing large data items to clients in anticipation of their profile requirements can often lead to inefficient use of network resources. Aperiodic push approaches can be converted to active rules without much difficulty.

3.2.2.1 Events

In data-driven processing, the primary activation events are modifications to affected data objects, either through data manipulation operations or by method invocations that alter object attributes. In addition, application errors or other exception events can trigger aperiodic push data transactions.

3.2.2.2 Conditions

Conditions for publish/subscribe mechanisms are evaluated to determine if new data modifications match client profiles. Profile management predicates are either individually optimized or merged with other individual profiles to permit efficient condition evaluation of rules.

3.2.2.3 Actions

In the simplest form, the action of an aperiodic push process could be a notification that is sent to a client whenever a data update occurs. Depending on the cache strategies in use, a rule action sequence may include a data manipulation operation for either a data object or object reference in a cached collection. With changes occurring to data objects, other possible actions that could be spawned are method calls for derived data computations or new query operations to refresh materialized views.

3.2.3 Periodic Pull

Periodic pull or polling mechanisms can be an effective way to disseminate information, if the right polling interval is used or the data update process is deterministic. Applications using polling can tolerate some staleness in data consistency in exchange for improved system throughput. Also, periodic pull can be used to retrieve heavyweight objects during non-peak times for network bandwidth-constrained environments. Finally, common broker nodes may effectively consolidate various client information needs by requesting data objects in a periodic manner. Active rules that are used for polling applications employ temporal triggers and actions are often view update operations.

3.2.3.1 Events

Activation events for periodic pull data processing are based on system clock interrupts. The time interrupt events usually occur on some regular interval depending on the data characteristics or the remote data update patterns of the application domain. A clock interrupt event may also occur as a relative time offset to the timestamp of some already signaled application event in polling applications.

3.2.3.2 Conditions

Similar to the aperiodic pull rules, the periodic pull data requests can evaluate the rule condition with an implied truth-value. The query constraint is a likely rule condition for retrieval requests. If the polling requests are conducted as batch updates during light processing periods, the condition predicate may also evaluate the presence of any data updates or assert whether the network utilization rate is under a certain threshold for heavyweight object dissemination.

3.2.3.3 Actions

Sub-query data retrieval actions are common for periodic pull rules in a hierarchical information structure. When requesting data objects or metadata, communication services are also valid actions for distributing results to clients. Finally, an important action for periodic data transactions is scheduling the clock interrupt event for next periodic processing cycle.

3.2.4 Periodic Push

Periodic push data dissemination, such as data broadcasting, transmits information on a regular cycle, and clients can monitor the broadcasts to retrieve relevant information. Clients subjected to occasional network down times can still recover missed information when periodic push dissemination is used. Additionally, periodic push data distribution relieves data servers from processing time-consuming data requests for numerous clients because users can regularly monitor broadcasts of frequently accessed database information. The challenges associated with data broadcasting are finding the right broadcast frequency and acquiring the communications bandwidth resources for dissemination. Periodic push rules use temporal events and data dissemination actions.

3.2.4.1 Events

Periodic push activation events are time-based system clock interrupts. With broadcast disk data dissemination, the client data access frequencies of database clusters is used to determine how often refreshed data should be distributed to the client community. Maintaining consistent push intervals is important for remote clients attempting to synchronize data monitoring activities with broadcast cycles.

3.2.4.2 Conditions

Periodic push-based rules use condition evaluation to match client profile information against the proposed data broadcast schedule. By maintaining an accurate client profile, data broadcasts may only need to transmit a subset of database objects each cycle. In addition, another relevant condition for periodic push processing may be verifying that data has changed prior to submitting a broadcast.

3.2.4.3 Actions

One common action for periodic push dissemination rules is invoking the communication routines for multicasting the data. Scheduling the next broadcast time trigger is also a necessary periodic push rule action. Finally, data updates to cached client collections may also be in the action sequence.

3.3 Active Rule Customization

The first two steps of this design methodology have extracted the essential dissemination-based application features and converted those data characteristics into basic rule components. The active rule customization process enhances the rule framework previously established by introducing additional rule elements to tailor rule-based reactive behavior to client needs. In this step of the methodology, variations of rule features that can impact the rule execution semantics are examined for the four data dissemination-based rule templates: aperiodic pull, aperiodic push, periodic pull, and periodic push.

Event-condition (E-C) coupling, condition-action (C-A) coupling, event consumption, rule granularities, and priorities are some of the active rule dimensions used for in the customization step. Since this methodology is based on generic active rule capabilities, not all features presented may appear in a specific active database implementation, but designers should consider which rule features must be supported as they make their implementation selection.

3.3.1 *Aperiodic Pull*

The miscellaneous active rule features for the aperiodic pull dissemination class are summarized in the following sections.

3.3.1.1 *Coupling*

Since asynchronous triggers are used to process 'on-demand' requests, immediate E-C coupling ensures rule conditions are promptly evaluated. For C-A coupling, detached coupling from the triggering transaction may be an effective performance choice for executing long duration sub-query transactions, but the failure of those separate transactions are independent of the main triggering transaction. However, immediate C-A coupling ensures prompt action execution as a dependent sub-transaction of the event-triggering transaction for add-hoc information requests.

3.3.1.2 *Priorities*

Unless rules are used in applications with firm real-time deadlines, the demand-driven aperiodic rules usually have higher relative priorities than time-based rules. Among rules in the same dissemination class, the information significance value can be used to determine the partial order of absolute rule priorities. For example, rules for lightweight objects may be preferred over rules governing heavyweight objects due to their performance advantage in a limited bandwidth environment.

3.3.1.3 *Miscellaneous Features*

For pull operations, set granularity is implied since the query request evaluates over all data instances. Chronicle event consumption mode ensures that each retrieval operation is handled in the order received.

3.3.2 *Aperiodic Push*

The coupling options, priorities, and other customization features are covered in the following sections for aperiodic push data transfers like the publish/subscribe mechanism.

3.3.2.1 *Coupling*

Immediate E-C and C-A coupling modes are most appropriate when instance-oriented push is in use since the nested sub-transactions for each data object need to complete promptly to avoid streamline transaction processing. For set-oriented processing, deferred E-C and C-A coupling allows all updates in a transaction to occur before checking the profile and distributing the results. Detached C-A coupling can be used for dispatching downstream derived-data computations to other data components.

3.3.2.2 *Priorities*

Among rules in the aperiodic push dissemination class, the information significance value should be the most important factor for determining absolute rule priorities. When data objects have similar information value, temporal data items with a shorter validity interval should possess a higher relative rule priority than objects with a longer data validity period. As a final consideration, rules for lightweight objects may be preferred over heavyweight objects due to the performance advantage.

3.3.2.3 *Miscellaneous*

For important or real-time data items, instance-level granularity ensures that transactions are processed immediately. However, performance considerations may dictate that frequent individual updates should be pooled and processed using set granularity. For non-temporal data, chronicle event consumption mode is the appropriate choice since update dependencies may exist between successive events. When multiple update events occur for temporal data, recent event consumption ensures that only the latest item with the longest data validity is used for rule processing.

3.3.3 *Periodic Pull*

When customizing periodic pull rules, developers must consider the execution features detailed in the following sections.

3.3.3.1 *Coupling*

Since periodic pull rule events may have tight timing constraints, immediate E-C coupling enables query conditions to be evaluated in an efficient manner. For C-A coupling, polling query actions should be executed using immediate coupling for prompt completion of time-based request. However, detached coupling can be used so longer duration upstream sub-queries of heavyweight objects can occur in separate transactions, allowing the triggering transaction to commit and release locks on data items. Regardless of which CA-coupling mode is used, the scheduling of the next periodic update should be the last action in the sequence.

3.3.3.2 *Priorities*

The information significance value should determine the precedence of rule execution priorities so transactions most important to a client are scheduled first. Since periodic pull transactions may be scheduled for off-hours updates, rules for heavyweight objects may be preferred over lightweight objects during light data traffic periods.

3.3.3.3 *Miscellaneous Features*

For periodic pull transactions, set granularity is implied as all relevant data instances are queried. To prevent starvation of time-based transactions, sequential execution of data retrievals using chronicle event consumption mode is appropriate for periodic polling.

3.3.4 *Periodic Push*

The miscellaneous active rule features for the periodic push dissemination class are summarized in the following sections.

3.3.4.1 *Coupling*

Immediate E-C and C-A coupling modes are most appropriate for periodic push transactions since remote clients may be using the data broadcast as their primary source of information. Deferred C-A coupling can be used for multicasting data to verify that all communications were received before scheduling the next push cycle.

3.3.4.2 *Priorities*

Due to the large volume of data transactions in dissemination applications, the information significance value should be the most important factor for determining rule execution priorities. For data broadcasts, the frequency of data distribution implicitly infers an execution priority for rules.

3.3.4.3 *Miscellaneous Features*

Set-oriented granularity is used for periodic push broadcasts almost by convention, since the entire database could be cyclically transmitted for some applications. Because broadcast schedules are potentially well synchronized with clients, chronicle event consumption mode would be the most appropriate choice.

3.4 Summary

The methodology steps presented in this chapter have formulated a process for incrementally transforming information distribution concepts into an active rule implementation approach. In the next chapter, the steps of the design methodology are applied to three test scenarios representative of a dynamic data application environment.

4 Case Study Analysis and Results

Chapter 3 presented a methodology to assist developers in the analysis and design of information dissemination concepts in the form of active database rules. This chapter outlines an application of that methodology to a test case by presenting the analysis and design of three nominal information dissemination scenarios from a dynamic data application, DARPA's Dynamic Database (DDB) research program.

The overarching goal of the DDB program [BBD+96] is to efficiently produce and continuously update a dynamic situation estimate of the evolving battlespace using all available sensor resources. The underlying information processing goals of DDB include: maintaining a comprehensive history of sensor data, generating newly derived information products from multiple data sources, and informing clients and other applications of significant database changes based on either pre-defined interests or data requests. The communications infrastructure and information dissemination services, used by DDB, would eventually be provided by another DARPA program, Battlefield Awareness and Data Dissemination (BADD) [DMS97][DSLL97][LS97][SDSV97].

In the rapidly changing data environment of DDB, there are many potential information push and information pull scenarios available for analysis in the case study. The first test case examined is a nominal information pull scenario that is representative of a DDB application process. Secondly, the methodology is applied to a plausible information push scenario from DDB. The final test case is a hybrid scenario involving both data push and pull technologies.

4.1 Analysis of Information Pull Scenario

In this scenario, a user wants to retrieve all data images in the database that contain an image number associated with a particular sensor. By reviewing a history of the stored images, the image analyst hopes to determine when the quality of images started to degrade and whether the sensor is malfunctioning. The client initiates a high priority data request to the sensor history database, which maintains a materialized collection of image references as shown in Figure 9.

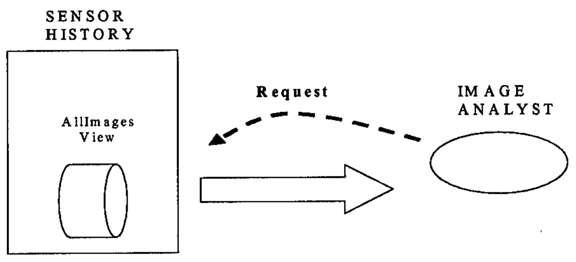

Figure 9: Information Pull Scenario

4.1.1 Application Features

The user-initiated request is not time-driven, so the activation mechanism in this scenario would be considered *data-driven*. The data request has an image number constraint, so the precision of the query results should be *exact*. The client wants to be able to examine the images, which are *heavyweight* objects. Finally, a *unicast* distribution is probably appropriate for this individual request. If the caching resources existed, multicast snooping of the images could take place from a common client cache area. Based on the analysis of the scenario application features, an *aperiodic pull* dissemination approach is recommended.

4.1.2 Rule Framework

The triggering event for this data scenario would be the user-directed query operation. The rule condition evaluates the image number query constraint over the image collection, AllImages. The first rule action stores the retrieved images in a transient storage collection, then a communications service is called to distribute the transient collection. Finally, the transient memory is recovered when the delete action occurs.

4.1.3 Customized Rules

Because this is an asynchronous request, immediate E-C coupling is used to quickly start the condition evaluation for this data request. However, deferred

C-A coupling is preferred, so that all images can be evaluated before any query results are returned. Since this request involves retrieving heavyweight objects and the information value is significant, the overall rule execution priority would probably be medium. Set-oriented granularity is assumed due to the data retrieval, and chronicle event consumption ensures the request is not preempted by a more recent request. Figure 10 shows a sample active rule structure for the DDB information pull scenario.

```
CREATE RULE  R1  FOR AllImages
ON     Client.query(image_number)
IF     [IMMEDIATE]
       AllImages.oid->GetNumber == image_number
THEN[DEFERRED]
       QueryResult.insert( DeRef( oid ) ); //DeRef gets image
       send( Client, QueryResult);
       delete QueryResult
PRIORITY   MEDIUM
```

Figure 10: Active Rule for Aperiodic Pull

4.2 Analysis of Information Push Scenario

Every six hours, new weather satellite images are processed. The current weather images are maintained in a sensor history database view, and new images are added to the view whenever updates occur. Since the weather images are so large, clients desiring the weather data must register their geographic location and network address with the server to receive the data. Because the images are considered a six-hour snapshot, the overall significance of the information is low in terms of real-time weather assessments. The scenario data flows are shown in Figure 11.

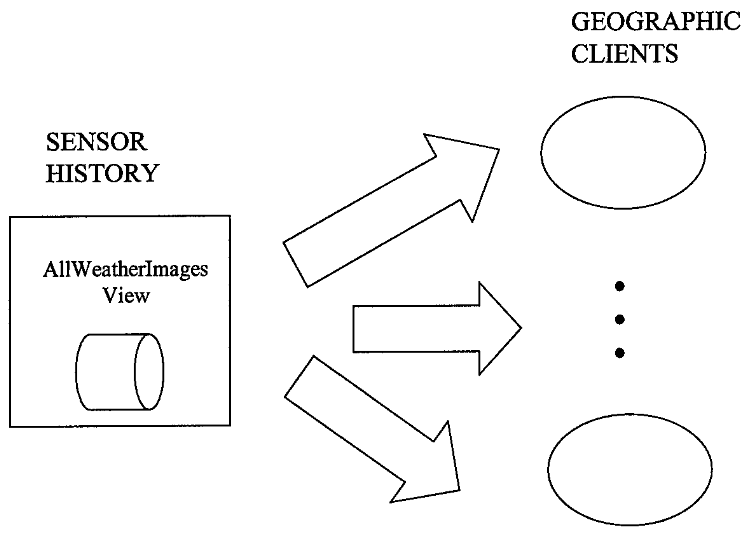

Figure 11: Information Push Scenario

4.2.1 *Application Features*

The weather source data is updated on a regular periodic basis, which implies that a *time-driven* activation mechanism would be the right choice. If only a small group of clients actually register for weather updates, current data is only disseminated for a subset of the image database, so client data precision would be considered *inexact*. The dissemination weather images indicates the data size factor is categorized as *heavyweight*. Several neighboring clients may require the same geographic images, so a *multicast* protocol is probably the best solution. A *periodic push* distribution mechanism is an appropriate solution for this scenario.

4.2.2 *Rule Framework*

The cyclic nature of the weather sensor updates suggests a clock-based interrupt as the rule-triggering event. The rule condition will perform the profile matching of registered client geographic location with the location of the image. The first rule action is the multicast transmission of images, and the next action is the scheduling of the next periodic interrupt.

4.2.3 *Customized Rules*

Immediate E-C is appropriate for this time-based triggered rule. Deferred C-A coupling will ensure the multicast transmission will reach all registered clients. Heavyweight objects with little information significance would lead to a low static rule priority, but the priority should increase dynamically as the image data validity deadline approaches. Set-oriented granularity coincides with the lone multicast distribution, and the temporal nature of the data suggests recent event consumption mode. With a six-hour window for processing these images, the chances of having multiple time interrupt events active is fairly remote. The proposed active rule for weather image periodic push is shown in Figure 12.

```
CREATE RULE  R2  FOR AllWeatherImages
ON     SixHourTimeUp()
IF     [IMMEDIATE]
       AllWeatherImages.oid->GetLoc INTERSECT
       Profiles.Client->GetLoc
THEN[DEFERRED]
       schedule( SixHourTimeUp, NOW + 6:00:00.00);
       multicast( DeRef( oid ),
         Profiles.Client->GetAddr ); //DeRef gets image
PRIORITY   LOW
```

Figure 12: Active Rule for Periodic Push

4.3 Analysis of Hybrid Dissemination Scenario

The commander client wants to know whenever an enemy unit is on the move with a certainty of at least 60 percent. The sensor history server maintains a view UnitMovers, which contains reference-pairs pointing to a entity (e.g., unit) and a track (e.g., road). The sensor history server checks both the entity and the tracks source data periodically. If a unit is moving on a road, a new object is inserted into the UnitMovers view, and if the object meets the certainty threshold, the commander client is notified immediately.

In this scenario, a hierarchical information structure is used where one component is both a consumer and producer of information as shown in Figure 13. As such, the overall data processing scenario is actually a compilation of two sub-processes: view update and client notification. An active rule structure will be created for each data dissemination sub-process.

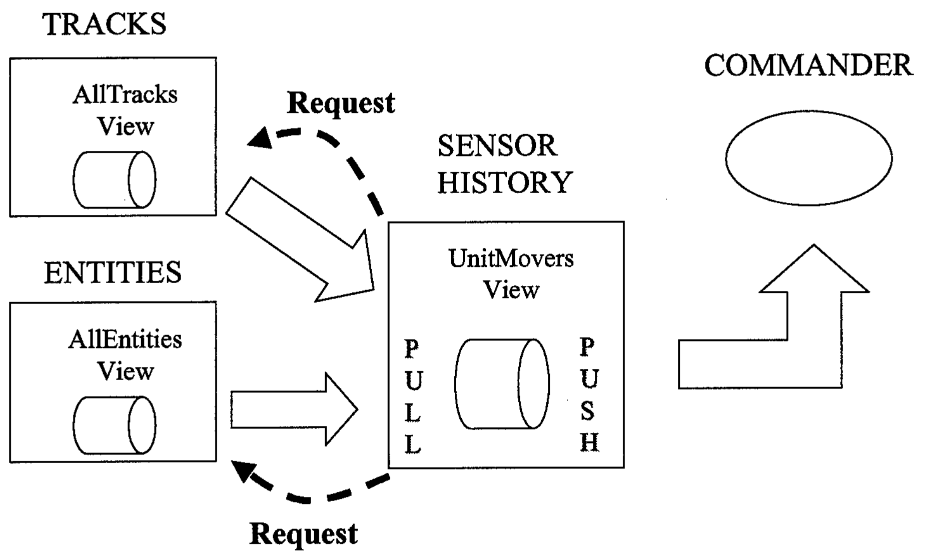

Figure 13: Push/Pull Hybrid Scenario

4.3.1 Application Features

For the view update, the activation mechanism is a *time-driven* interrupt, which for this process is every minute. The entity and track source information must be accurate and meet the query constraints, so data precision is *exact*. Only data references are used in the updates; therefore, *lightweight* is the data size feature. Lastly, a *unicast* connection can be used to distribute updates. Based on the analysis of the scenario application features, a *periodic pull* dissemination approach is recommended.

On the other hand, the client notification process uses a *data-driven* trigger whenever view modifications occur. *Inexact* data precision may apply here since the client may be notified as the certainty level nears the threshold, but has not reached the constraint level. The data size is *lightweight* message dissemination, and the sensitivity of information warrants a *unicast* communication connection. An *aperiodic push* distribution mechanism is an appropriate solution for this scenario.

4.3.2 Rule Framework

The view update rule needs a temporal interrupt event every minute. The rule condition is the query join constraint of entities and tracks. The scheduling of the next interrupt is the first action, and the insert or update operation on the view is the second action.

For the client notification rule, the view data manipulation operation is the event trigger. The profile match conditions are included in the notification rule condition. The only action is the alert notification to the client.

4.3.3 Customized Rules

With the temporal characteristics of the entity and track data, the E-C and C-A coupling modes for the view update rule should be immediate. The high information significance and time-based data validity warrants a high rule priority. The periodic query implies set granularity, and recent event consumption ensures timely refreshing of temporal data. The active rule format of the view update sub-process is shown in Figure 14.

```
CREATE RULE  R3  FOR UnitMovers
ON     MinuteTimeUp()
IF     [IMMEDIATE]
       AllEntities.oid->GetUnitID ==
           AllTracks.oid->GetTUnitID
       AND AllEntities.oid->GetSide == 'ENEMY'
       AND AllTracks.oid->GetAction == 'MOVING'
THEN[IMMEDIATE]
       schedule(MinuteTimeUp, NOW + 00:01:00.00);
       insert(AllEntities.oid, AllTracks.oid );
PRIORITY   HIGH
```

Figure 14: Active Rule for Periodic Pull

For the commander notification, the E-C and C-A coupling modes are deferred to allow all view changes in the transaction to occur before checking the uncertainty profile condition. The rule priority is high because of the significance of the information value. Instance-oriented granularity would work for this rule if you sent the alert on every view update, but since deferred C-A coupling is used, the rule granularity is set-oriented. Chronicle event consumption makes sense for this asynchronous process. The client notification active rule is shown in Figure 15.

```
CREATE RULE  R4  FOR UnitMovers
ON     insert()
IF     [DEFERRED]
       certainty >= UMProfiles.Client->GetConfLvl
THEN[DEFERRED]
       alert( "Enemy Units Moving!", Profiles.Client->GetAddr );
PRIORITY   HIGH
```

Figure 15: Active Rule for Aperiodic Push

4.4 Summary of Results

By applying the methodology to these three test case scenarios, active rules were designed for all four of the dissemination-based classes. Although some active rule dimensions were not exercised in these test cases, the scenarios did effectively demonstrate the use of the information dissemination-based active rule methodology for nominal design tasks. To determine the methodology's utility for more robust design tasks, test cases exploring some of the more esoteric features of active rule design would be necessary.

5 Conclusions

The immense volume of transactions involved in dynamic data environments requires flexible control structures to effectively balance the flow of information between producers and consumers. As one promising solution, dissemination-based information systems provide a scalable variety of distribution mechanisms using both data push and pull technologies. Even closer to the underlying data, active database rules can embed reactive behavior, normally found in application code, right into the numerous transactions occurring within a dynamic data environment. Capturing the capabilities of data dissemination systems in the form of active rules for more effective dynamic database control was the goal of this research effort.

The significant contribution of this research is the introduction of a design methodology that evolves information distribution concepts of an application into a consistent form closely coupled to the affected data. The methodology presented is divided into a dissemination-based application analysis phase, an active rule transformation phase, and a rule customization phase. The three-step methodology was successfully demonstrated using four different dissemination approaches in a dynamic data case study.

5.1 Findings

The four data features used to assess the information dissemination application were effective not only with classifying a data delivery strategy, but also with introducing design considerations such as client caching and communications protocols. However, analyzing a database application from the viewpoint of distribution principles in this methodology proved to be a bit orthogonal from traditional database design techniques.

Building an active rule framework from a data dissemination class was relatively straightforward, although each rule event, condition, or action usually had multiple options to consider. The versatility of the E-C-A rule model enables many behavioral alternatives to exist based on the configuration of those three rule components. Without any formal rule analysis included in the methodology, the inherent rule flexibility could eventually lead to rule execution problems for designers.

The rule customization step was difficult to perform from an abstract design perspective. Some of the rule execution features are configurable, like coupling modes, but rule granularity and event consumption modes can sometimes be implementation-dependent features. Ideally, this step could be automated, and only features configurable in the target implementation rule language would require designer input.

The case studies successfully demonstrated the methodology on some simple, but representative, dynamic data test cases. The test scenarios were selected to illustrate the application of the design process on all four classes of dissemination. The testing of the methodology using more sophisticated active rule features such as composite events and detached coupling is left as future work.

5.2 Recommendations

As a result of the knowledge obtained from this research, the following recommendations for the application and extension of the dissemination-based active rule methodology are presented.

5.2.1 Assess DDB for Active Database Selection

Technological forecasts for future contractual phases of the Dynamic Database program include the integration of active database technology with the initial DDB demonstration system to provide more flexible control of data computations. By assessing the DDB functionality using the proposed methodology, designers can decide which active database dimensions they are likely to need and select an active database system that supports those features.

5.2.2 DDB Development Process Integration

To fully realize all DDB design goals, the steps of the active dissemination-based methodology should be at least compatible, if not fully integrated, with the overall application design process used by DDB team members. The integration effort would involve introducing active rule design concepts among the use-case scenarios and schema development activities of DDB.

5.2.3 Technology Bridge for DDB and BADD

The dissemination-based active rule structures, developed as part of this research, provide an excellent transition mechanism between two key components of DARPA's battlefield awareness architecture. The Dynamic Database (DDB) program maintains the active repository of information products, and the Battlefield Awareness and Data Dissemination (BADD) program provides the intelligent data dissemination services for warfighting clients. By considering the information distribution properties as part of the active database design activity, the application data flows can seamlessly migrate from the storage component to the dissemination mechanism.

5.3 Future Areas of Research

There are several areas of related research that are relevant to the active dissemination-based design methodology. Future research areas include:

adding formal rule analysis to the methodology, automating the rule design tool, and experimenting with dynamic dissemination performance conditions.

5.3.1 *Integrate Rule Analysis into Methodology*

The formal properties of active rules can be statically analyzed to predict rule execution behavior. Rule analysis research has examined termination, confluence, and observable determinism properties for active rules [AHW95][FT95][KC95][VGD97]. In addition, rule grouping design strategies called stratification [BCP96][CF97] could be added to the methodology. Including rule analysis as part of this methodology would make the dissemination-based active rule design process much more robust.

5.3.2 *Automate Methodology Design Tool*

An automated design tool based upon the dissemination-based active rule methodology would be a valuable asset for a developer. The steps of the methodology could be presented to the rule designer in a graphical format, and a repository of rules and templates would also make rule maintenance easier. Finally, the design tool could produce rule constructs in a target rule language for active databases.

5.3.3 Dynamic Push/Pull Rule Condition Experiments

The methodology proposed in this research statically determines the information dissemination approach. Further research could examine the essential conditions for determining data distribution performance [AKGM96] so active rules could use different dissemination strategies based on a performance condition. Also, collaborative intelligent agent research may reveal effective methods for dynamically adjusting information flow between data nodes.

5.4 Summary

While the active database rule design techniques applied in this research have provided a contribution to information dissemination systems development, the constant growth of widespread dynamic data applications will continue to challenge researchers to find scalable and flexible solutions to manage high-velocity information flow among clients. This research effort facilitates the migration of design principles between both the information dissemination and the active database fields.

Bibliography

[AC95] E. Anwar, S. Chakravarthy, "Realizing Transaction Models: An Extensible Approach using ECA Rules," *Technical Report UF-CIS-TR-95-029*, University of Florida, 1995.

[AF98] D. Aksoy, M. Franklin, "Scheduling for Large-Scale On-Demand Data Broadcasting," In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'98)*, March 1998.

[AFZ95] S. Acharya, M. Franklin, S. Zdonik, "Dissemination-Based data Delivery Using Broadcast Disk," *IEEE Personal Communications*, pp. 50-60, December 1995.

[AFZ96] S. Acharya, M. Franklin, S. Zdonik, "Prefetching from a Broadcast Disk," In *Proceedings of the Twelfth International Conference on Data Engineering (ICDE'96)*, pp. 276-285, March 1996.

[AFZ97] S. Acharya, M. Franklin, S. Zdonik, "Balancing Push and Pull for Data Broadcast," In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pp. 183-194, May 1997.

[AHW95] A. Aiken, J. Hellerstein, J. Widom, "Static Analysis Techniques for Predicting the Behavior of Active Database Rules," *ACM Transactions on Database Systems*, pp. 3-41, March 1995.

[AKGM96] B. Adelberg, B. Kao, H. Garcia-Molina, "Database Support for Efficiently Maintaining Derived Data," In *Proceedings of the Fifth International Conference on Extending Database Technology (EDBT '96)*, pp. 223-240, March 1996.

[BB97] S. Baruah, A. Bestavros, "Pinwheel scheduling for Fault-Tolerant Broadcast Disks in Real-time Database Systems," In *Proceedings of the Thirteenth International Conference on Data Engineering (ICDE'97)*, pp. 543-551, April 1997.

[BBD+96] P. Baim, B. Bolles, D. DeWitt, et al., "Concepts for a Dynamic Database," unpublished presentation, September 1996. Available at http://dtsn.darpa.mil/iso.

[BCL98] M. Berndtsson, S. Chakravarthy, B. Lings, "Extending Active Capability Mechanisms for Context Based Subscriptions," *Technical Report HS-IDA-TR-98-007*, University of Skovde, November 1998.

[BCP96] E. Baralis, S. Ceri, S. Paraboschi, "Modularization Techniques for Active Rules Design," *ACM Transactions on Database Systems*, pp. 1-29, March 1996.

[Buc98] A. Buchmann, "Architecture of Active Database Systems," In N. Paton, *Active Rules in Database Systems*, Springer-Verlag, 1998.

[BZBW95] A. Buchmann, J. Zimmerman, J. Blakeley, D. Wells, "Building an Integrated Active OODBMS: Requirements, Architecture, and Design Decisions," In *Proceedings of the Eleventh International Conference on Data Engineering (ICDE'95)*, pp. 117-128, March 1995.

[CA95] S. Chakravarthy, E. Anwar, "Exploiting Active Database Paradigm for Supporting Flexible Transaction Models," *Technical Report UF-CIS-TR-95-026*, University of Florida, April 1995.

[CB97] G. Cybenko, B. Brewington, "The Foundations of Information Push and Pull," to appear in D. O'Leary (ed.), *Proceedings of the IMA Workshop on the Mathematics of Information*, Springer-Verlag, September 1997.

[CBGM98] A. Crespo, O. Buyukkokten, H. Garcia-Molina, "Efficient Query Subscription Processing in a Multicast Environment," unpublished technical report, Stanford University, July 1998.

[CF97] S. Ceri, P. Fraternali, *Designing Database Applications with Objects and Rules: The IDEA Methodology*, Addison-Wesley, 1997.

[Cha92] S. Chakravarthy, "Architectures and Monitoring Techniques for Active Databases: An Evaluation," *Technical Report UF-CIS-TR-92-041*, University of Florida, 1992.

[CJL91] M. Carey, R. Jauhari, M. Livny, "On Transaction Boundaries in Active Databases: A Performance Perspective," *IEEE Transactions on Knowledge and Data Engineering*, pp. 320-336, September 1991.

[CR96] S. Ceri, R. Ramakrishnan, "Rules in Database Systems," *ACM Computing Surveys*, pp.109-111, March 1996.

[DBB+88] U. Dayal, B. Blaustein, A. Buchmann, U. Chakravarthy, M Hsu, R. Ledin, D. McCarthy, A. Rosenthal, S. Sarin, "The HIPAC Project: Combining Active Databases and Timing Constraints," *SIGMOD Record*, pp. 51-70, March 1988.

[DDFA98] "The Dynamic Database: Functional Architecture (Version 0.32)," unpublished document, October 1998.

[DFJ+96] S. Dar, M. Franklin, B. Jonsson, D. Srivastava, M. Tan, "Semantic Data Caching and Replacement," In *Proceedings of 22nd International Conference on Very Large Data Bases (VLDB'96)*, pp. 330-341, September 1996.

[DG93] K. Dittrich, S. Gatziu, "Time Issues in Active Database Systems," In *Proceedings of the 1st International Workshop on an Infrastructure for Temporal Databases*, June 1993.

[DGG96] K. Dittrich, S. Gatziu, A. Geppert (eds.), "The Active Database Management System Manifesto: A Rulebase of ADBMS Features," *SIGMOD Record*, pp. 40-49, September 1996.

[DHL90] U. Dayal, M. Hsu, R. Ladin, "Organizing Long-Running Activities with Triggers and Transactions," In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pp. 204-214, May 1990.

[DMS97] R. Douglass, J. Mork, B.R. Suresh, "Battlefield Awareness and Data Dissemination (BADD) for the Warfighter," In *Proceedings of the SPIE*, Volume 3080, pp. 18-24, April 1997.

[DSLL97] J. Dukes-Schlossberg, Y. Lee, "IIDS: Intelligent Information Dissemination Server," In *Proceedings of the 1997 IEEE Military Communications Conference (MILCOM'97)*, pp. 635-639, November 1997.

[FT95] P. Fraternali, L. Tanca, "A Structured Approach for the Definition of the Semantics of Active Databases," *ACM Transactions on Database Systems*, pp. 414-471, December 1995.

[FWP97] A. Fernandes, M. Williams, N. Paton, "A Logic-Based Integration of Active and Deductive Databases," *New Generation Computing*, pp. 205-244, 1997.

[FZ96] M. Franklin, S. Zdonik, "Dissemination-Based Information Systems," *IEEE Data Engineering Bulletin*, pp. 19-28, September 1996.

[FZ97] M. Franklin, S. Zdonik, "A Framework for Scaleable Dissemination-Based Systems," In *Proceedings of the 1997 ACM SIGPLAN Conference on Object Oriented Programming Languages and Applications (OOPSLA'97)*, pp. 94-105, October 1997.

[FZ98] M. Franklin, S. Zdonik, "'Data in Your Face': Push Technology in Perspective," In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pp. 516-519, June 1998.

[GD93] S. Gatziu, K. Dittrich, "Events in an Active Object-Oriented Database System," In *Proceedings of the 1st International Workshop on Rules In Database Systems (RIDS'93)*, pp. 23-39, September 1993.

[GFV96] S. Gatziu, H. Fritschi, A. Vaduva, "SAMOS an Active Object-Oriented Database System: Manual," *Technical Report 96.02*, University of Zurich, February 1996.

[GGD+95] A. Geppert, S. Gatziu, K. Dittrich, H. Fritschi, A. Vaduva, "Architecture and Implementation of the Active Object-Oriented Database Management System SAMOS," *Technical Report 95.29*, University of Zurich, November 1995.

[GKVB+98] S. Gatziu, A. Koschel, G. von Bultzingsloewen, H. Fritschi, "Unbundling Active Functionality," *SIGMOD Record*, pp. 35-40, March 1998.

[Gla96] D. Glance, "Multicast Support for Data Dissemination in OrbixTalk," *IEEE Data Engineering Bulletin*, pp. 29-36, September 1996.

[HAC+97] E. Hanson, N. Al-Fayoumi, C. Carnes, M. Kandil, H. Liu, M. Lu, J.B. Park, A. Vernon, "TriggerMan: An Asynchronous Trigger Processor as an Extension to an Object-Relational DBMS," *Technical Report TR-97-024*, University of Florida, December 1997.

[HBC+97] E. Hanson, S. Bodagala, U. Chadaga, M. Hasan, G. Kulkarni, J. Rangarajan, "Optimized Trigger Condition Testing in Ariel using Gator Networks," *Technical Report TR-97-002*, University of Florida, February 1997.

[HNK94] J. Han, S. Nishio, H. Kawano, "Knowledge Discovery in Object-Oriented and Active Databases," In F. Fuchi, T. Yokoi, *Knowledge Building and Knowledge Sharing,* Ohmsha Ltd. and IOS Press, pp. 221-230, 1994.

[KC95] S.K. Kim, S. Chakravarthy,, "A Confluent Rule Execution Model for Active Databases," *Technical Report UF-CIS-TR-95-032,* University of Florida, October 1995.

[LEF98] Q. Lu, M. Eichstaedt, D. Ford, "Efficient Profile Matching for Large Scale Webcasting," *Computer Networks and ISDN Systems,* pp. 443-455, April 1998.

[LRST93] P. Loborg, T. Risch, M. Skold, A. Torne, "Active Object-Oriented Databases in Control Applications," *Technical Report LiTH-IDA-R-93-28,* Linkoping University, 1993.

[LS97] M. Lazaroff, P. Sage, "Any Information, Anywhere, Anytime, for the Warfighter," In *Proceedings of the SPIE,* Volume 3080, pp. 35-42, April 1997.

[MT95] D. Montesi, R. Torlone, "A Transaction Transformation Approach to Active Rule Processing," In *Proceedings of the Eleventh International Conference on Data Engineering (ICDE'95),* pp. 109-116, March 1995.

[NTM+95] S. Navathe, A. Tanaka, R. Madhavan, Y.H. Gan, "A Methodology for Application Design Using Active Database Technology," *Technical Report RL-TR-95-41,* Air Force Contract F30602-93-C-0175, Rome Laboratory, March 1995.

[Pat95] N. Paton, "Supporting Production Rules Using ECA-Rules in an Object-Oriented Context," *Information and Software Technology,* pp. 691-699, 1995.

[PDW+93] N. Paton, O. Diaz, M.H. Williams, J. Campin, A. Dinn, A. Jaime, "Dimensions of Active Behaviour," In *Proceedings of the 1st International Workshop on Rules In Database Systems (RIDS'93),* pp. 40-57, September 1993.

[PSS93] B. Purimetla, R. Sivasankaran, J. Stankovic, "A Study of Distributed Real-Time Active Database Applications," *IEEE Workshop on Parallel and Distributed Real-Time Systems,* April 1993.

[RS98] R. Ramakrishnan, A. Silberschatz, "Scalable Integration of Data Collections on the Web," *Technical Report CS-TR-98-1376*, University of Wisconsin, Madison, 1998.

[RSS+96] K. Ramamritham, R. Sivasankaran, J. Stankovic, D. Towsley, M. Xiong, "Integrating Temporal, Real-Time, and Active Databases," *SIGMOD Record*, pp. 8-12, March 1996.

[SDSV97] T. Stephenson, B. DeCleene, G. Speckert, H. Voorhees, "BADD Phase II: DDS Information Management Architecture," In *Proceedings of the SPIE*, Volume 3080, pp. 49-58, April 1997.

[SST96] R. Sivasankaran, J. Stankovic, D. Towsley, B. Purimetla, K. Ramamritham, "Priority Assignment in Real-Time Active Databases," *VLDB Journal*, pp.19-34, January 1996.

[SUC98] Y. Saygin, O. Ulusoy, S. Chakravarthy, "Concurrent Rule Execution in Active Databases," *Information Systems*, pp. 39-64, January 1998.

[TGNO92] D. Terry, D. Goldberg, D. Nichols, B. Oki, "Continuous Queries over Append-Only Databases," In *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data*, pp. 321-330, June 1992.

[Ulu98] O. Ulusoy, "Transaction Processing in Distributed Active Real-Time Database Systems," *Journal of Systems and Software*, pp. 247-262, September 1998.

[VGD97] A. Vaduva, S. Gatziu, K. Dittrich, "Investigating Termination in Active Database Systems with Expressive Rule Languages," *Technical Report 97.03*, University of Zurich, April 1997.

[WC96] J. Widom, S. Ceri, *Active Database Systems: Triggers and Rules for Advanced Database Processing*, Morgan Kaufmann Publishers, 1996.

[Wid93] J. Widom, "Deductive and Active Databases: Two Paradigms or Ends of a Spectrum?" In *Proceedings of the 1st International Workshop on Rules In Database Systems (RIDS'93)*, pp. 306-315, September 1993.

[Wid96] J. Widom, "Starburst Active Database Rule System," *IEEE Transactions on Knowledge and Data Engineering*, pp. 583-595, August 1996.

[XSR+96] M. Xiong, J. Stankovic, K. Ramamritham, D. Towsley, R. Sivasankaran, "Maintaining Temporal Consistency: Issues and Algorithms," *First International Workshop on Real-Time Databases*, March 1996.

[YA95] T.W. Yan, J. Annevelink, "A Powerful Wide-Area Information Client," In *Proceedings of the 1995 IEEE Computer Conference (COMPCON'95)*, pp. 13-18, March 1995.

[YGM94] T.W. Yan, H. Garcia_Molina, "Distributive Selective Dissemination of Information," In *Proceedings of the Third International Conference on Parallel and Distributed Information Systems (PDIS'95)*, pp. 89-98, September 1994.

[YGM95] T.W. Yan, H. Garcia_Molina, "SIFT – A Tool for Wide-Area Information Dissemination," In *Proceedings of the 1995 USENIX Technical Conference*, pp. 177-186, January 1995.

Vita

Captain Robert H. Hartz was born on 5 August 1962 in Ashland, Pennsylvania. In 1980, he graduated from North Schuylkill High School of Ashland, and shortly thereafter, he enlisted in the Air Force as an inventory management specialist. After assignments in supply at Eielson AFB, Alaska, and Grissom AFB, Indiana, he retrained into the computer-programming field in 1985. He was subsequently assigned to Detachment 1, 4200 Test and Evaluation Squadron, Castle AFB, California as a maintenance programmer for the KC-135 Digital Aircrew Training Devices. In 1987, he was selected for the Airman's Education and Commissioning Program and attended Arizona State University as a technical sergeant. In 1990, he graduated cum laude with a Bachelor's of Science in Engineering degree in the Computer Systems Engineering program.

In 1991, Capt Hartz graduated from Officer Training School and received a commission as a second lieutenant. He was initially assigned as Development Computer Engineer for the 3302nd Technical Training Squadron, Keesler AFB, Mississippi, and then he served as element chief of the 81st Communications Squadron's Software Support Flight in 1992. In 1994, he was reassigned to Armstrong Lab's Logistics Research Division, Wright-Patterson AFB, Ohio, as Computer Engineering Research Officer. He completed Squadron Officer School in-residence in 1996, and he began attending the Air Force Institute of Technology in 1997. After graduation from AFIT in 1999, Capt Hartz will be assigned to the Air Force Operational Test and Evaluation Center's Software Analysis Team, Kirtland AFB, New Mexico.